

**REMARKS / ARGUMENTS**

In the present Office action, dated April 23, 2004, a 3-way restriction requirement was made under 35 U.S.C. § 121 as set forth below:

Group I: Claims 1 – 14, 27, and 30;

Group II: Claims 15 – 26, 29, 31, and 32;

Group III: Claim 28.

Pursuant to an April 15, 2004 telephone conference between Examiner Chen and one of Applicants' representatives, Mr. John B. Conklin, an election was made, with traverse, to pursue the claim set in Restriction Group III.

In the application, no claims currently stand allowed and claim 28 stands rejected. Claim 28 was rejected under 35 U.S.C. § 102(b) as being anticipated by Sonderegger ("Sonderegger"), U.S. Patent No. 5,692,129.

Applicants have amended claim 28 to include a limitation of "a list of orphaned objects tested in accordance with a specifiable policy." *See* Application, pg. 23, ll. 18 – 23, pg. 24, ll. 1, 14, 15. Furthermore, Applicants have added independent claim 40 and dependent claims 33 – 39 and 41 – 48. Finally, as required by the present Office action, Applicants have also amended the Abstract. No new matter has been introduced by these amendments.

In view of the amendments and remarks herein, it is respectfully requested that a rejection of claim 28 be reconsidered and withdrawn.

***Sonderegger Does Not Teach Testing a List of Orphaned Objects In Accordance With a Specifiable Policy***

As noted above, the Office action rejects claim 28 under 35 U.S.C. § 102(b) as anticipated by the Sonderegger reference. The Sonderegger reference teaches a method and apparatus for management of application programs in a computer network through the use of a hierarchical database which includes application objects representing applications and their execution environments. *See* Sonderegger, Abstract. The management is accomplished through the use of a database coupled with “administrator tools” and an “application launcher.” *Id.* The “administrator tools” are used to support the creation, deletion, and modification of application objects. *Id.* The “application launcher” queries the database and updates a list of available applications kept on the user’s desktop, as well as automatically launches specified applications and cleans up the execution environment after an application terminates. *See* Sonderegger, Abstract, col. 21, ll. 46 – 48.

Independent claim 28, on the other hand, relates to a system for managing a directory of published objects, the directory maintained on a computer network, and removing nonfunctional objects. *See* Application, pg. 14, ll. 12 – 17, pg. 23, ll. 10 – 16. Specifically, the system of claim 28 contains a pruning module, a domain controller hosting the pruning module and the directory of published objects, as well as a thread for executing the pruning module. Furthermore, as amended in this response, the pruning module of claim 28 contains a further limitation in a form of a “module for determining if

the specified object is deleteable, wherein **a list of orphaned objects is tested in accordance with a specifiable policy.**” This limitation is not present in the Sonderegger reference.

A. *“Application Launcher” Does Not Teach Testing a List of Orphaned Objects  
In Accordance With a Specifiable Policy*

The Office action asserts that the Sonderegger reference teaches the above limitation by stating that Sonderegger discloses a “module for determining if the specified objects is deleteable [e.g., the updating or final cleaning module of the application launcher, col. 3, lines 24-42].” *See* Office action, pg. 4.

However, it is clear that Sonderegger does not teach a specifiable policy which is used to test a list of orphaned objects prior to a determination whether a specified object is deleteable. Instead, Sonderegger discloses an “application launcher” which is used to automatically launch network applications and to set up the application environment based on application objects queried from the database. *See* Sonderegger, col. 7, ll. 20 – 28. The application launcher is also used to “clean up” the application environment after the program is closed by the user, as well as to “update” network application icons displayed on a user’s desktop. *See* Sonderegger, col. 3, ll. 24 – 48. However, the “clean up” and desktop icon “update” functions of the application launcher in Sonderegger do not involve a creation of a list of orphaned objects which is processed according to a specifiable policy, as required by the present invention.

With regards to the “update” functionality of the application launcher, Sonderegger teaches that the desktop icon “update” function occurs *every time* a new application object is either added to or deleted from a database of application objects by the administrator. *See* Sonderegger, col. 19, ll. 19 – 23. Specifically, in discussing the manual and automatic icon update functionalities of the application launcher, Sonderegger states that “both of the steps 144 [user-initiated update] and 156 [automatic update] are performed to reduce the workload of the network administrator by reducing or eliminating the need for the administrator to manually update user desktops after changing the application objects in the schema database...” *Id.* Thus, no mention is made of a creation of a list of orphaned objects which is tested according to a specifiable policy *prior* to making a determination of whether a given object needs to be “updated”. Rather, such “updates” *strictly* correspond to and track any updates of application objects within the application object database. The present invention, on the other hand, requires that upon a determination that a given object may have become “orphaned” (e.g., a device which the object represents is no longer functional), a list of such objects is created and compared to a specifiable deletion policy prior to actually deleting the object from the database or directory of such objects. *See* Application, pg. 24, ll. 13 – 16.

Similarly, with regards to the “clean up” functionality of the application launcher, Sonderegger teaches that “the launcher cleans up by unmapping drives, releasing captured ports, and detaching from servers as needed.” *See* Sonderegger, col. 3, ll. 44 – 47. However, Sonderegger does not mention or imply that such “clean up” is performed according to a previously-created list of objects which may have become orphaned, nor

does Sonderegger mention or imply that a policy is checked prior to performing the “clean up” step.

Therefore, the “application launcher” of Sonderegger does not teach a necessary limitation of this invention, which requires testing a list of orphaned objects in accordance with a specifiable policy in order to determine whether an object is deleteable.

*B. “Snap-in Module” Does Not Teach Testing a List of Orphaned Objects In Accordance With a Specifiable Policy*

Finally, the Office action asserts that Sonderegger teaches the pruning module of this invention through its disclosure of a “snap-in module.” *See* Office action, pg. 4. The “snap-in module” of Sonderegger is part of the “administrator tools” used to create, delete, and modify the application objects and desktop attributes. *See* Sonderegger, col. 7, ll. 11 – 15, 41, 42. However, similar to the “application launcher” functionality of Sonderegger, the “snap-in module” does not expressly or inherently teach the use of an orphaned objects list which is tested in accordance with a specifiable policy prior to deletion of an object. Instead, the “snap-in module” *relies on the administrator to initiate* the creation, modification, or deletion of the “application objects available to a given user, group, or container class.” *See* Sonderegger, col. 7, ll. 53 – 57. In fact, the “snap-in module” functionality of Sonderegger teaches away from any use of a specifiable policy to be used to test a list of orphaned objects prior to their deletion since some of the routines used in the “snap-in module” require an *actual confirmation by the administrator*

prior to their execution. *See* Sonderegger, col. 9, ll. 31 – 34. For example, the “SnapinDesktopProc()” routine of the “snap-in module” “takes care to identify aliases and to allow changes to desktop attributes on desktop objects 51 in the database 38 *only after confirmation by the administrator.*” *Id.* (emphasis added).

Therefore, the “snap-in module” of Sonderegger does not teach a necessary limitation of this invention, which requires testing a list of orphaned objects in accordance with a specifiable policy in order to determine whether an object is deleteable.

Consequently, the Sonderegger reference does not, explicitly or implicitly, teach all the elements of claim 28. Accordingly, favorable reconsideration of claim 28 is respectfully requested. All of the pending independent claims now include language clarifying that the list of orphaned objects is tested in accordance with a specifiable policy node or depend from claims that include such language. Therefore, Applicants submit that the rejection in the Office action has been rendered moot. As this technique for pruning orphaned objects from a set of published objects within a directory service or other database is not discussed in, nor rendered obvious by, the cited art, Applicants further submit that all of the pending claims are now allowable.


### **CONCLUSION**

In view of the above amendments and remarks, the application is considered to be in good and proper form for allowance, and the Examiner is respectfully requested to pass this application to issue. If, in the opinion of the Examiner, a telephone conference would

In re Appln. of WILSON et al.  
Application No. 09/527,309  
Reply to Office action of April 23, 2004

expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney.

Respectfully submitted,



---

Scott H. Schulhof, Reg. No. 53,568  
One of the Attorneys for Applicant  
LEYDIG, VOIT & MAYER, LTD.  
Two Prudential Plaza, Suite 4900  
180 North Stetson  
Chicago, Illinois 60601-6780  
(312) 616-5600 (telephone)  
(312) 616-5700 (facsimile)

Date: August 23, 2004